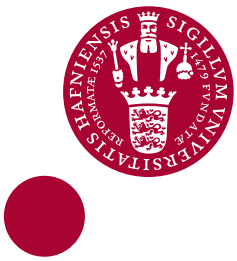


RoMo

avoiding conflicts between the physical and digital model in tabletop interfaces with robotic tangibles

Pedersen, Esben Warming; Hornbæk, Kasper

Published in:
Proceedings of the 11th Danish Human-Computer Interaction Research Symposium

Publication date:
2011

Document version
Peer reviewed version

RoMo: Avoiding Conflicts Between the Physical and Digital Model in Tabletop Interfaces with Robotic Tangibles

Esben Warming Pedersen & Kasper Hornbæk
Department of Computer Science, University of Copenhagen
DK-2300 Copenhagen S, Denmark
{esbenwp, kash}@diku.dk

ABSTRACT
In TUIs, physical/digital conflicts can occur when the digital model does not match the model implied by the spatial layout of tangibles. We show how tangible tabletop interfaces (TTI) can be modified to allow robot movement of tangibles, thereby avoiding conflicts. We present RoMo, an open source Java library that allow existing TTI applications to perform robot movement, and demonstrate its functionality with three applications.

Author Keywords
Tangible user interfaces, bidirectional user interfaces, robotic tangibles

ACM Classification Keywords
H.5.2. User Interfaces. D.2.2. Software libraries.

INTRODUCTION
In a landmark paper in the field of tangible user interfaces (TUIs), Ishii and Ullmer described the risk of physical/digital conflicts [3]. Physical/digital conflicts emerge when the digital model does not match the model implied by the spatial layout of tangibles. Conflicts can occur if the digital model is dynamically changing (i.e., if it depends on other data-sources than user input) or if multiple users are collaborating on TUIs in different places sharing the same digital model. Physical/digital conflicts destroy the perception of input/output unification and may lead to ambiguous interpretations of the tangibles' spatial layout.

To address the problem of physical/digital conflicts in the field of tabletop tangible interfaces (TTIs), researchers have developed bidirectional tangible interfaces that are capable of moving tangibles around the tabletop surface. In contrast to unidirectional tangible interfaces, this capability allows bidirectional user interfaces to use tangibles as output devices and display changes in the digital model physically. The bidirectional tangible interfaces may be divided into two groups: (a) interfaces with small unpowered pucks, moved by an electromagnetic array situated under the tabletop surface [1, 5, 6], and (b) interfaces with battery-powered robots that move on the tabletop surface [7–10, 12]. Most notable is PICO [6], an interface that uses magnetic attraction and repulsion as haptic user guidance, thereby allowing the computer and the user to collaborate in solving optimization tasks.

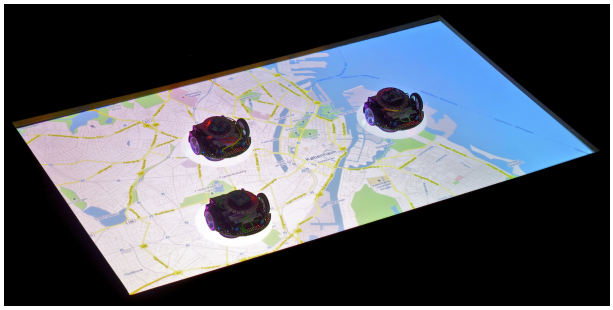


Figure 1. A map application using RoMo to move three robotic tangibles.

Whereas existing bidirectional tangible interfaces in part preclude physical/digital conflicts, they introduce other problems. The systems that employ electromagnetic arrays use custom-made hardware to move the pucks, and building them requires advanced engineering skills. The number of magnets and control boards required to move the pucks grow with the size of the tabletop, making these interfaces very expensive. The systems with battery-powered robots require far less hardware and are easier to build. However, existing systems rely on custom-made tracking and moving software which makes it impossible for other systems to adopt their robot control technologies.

The present paper makes two contributions to the field of tangible tabletop interfaces. First, we describe the hardware setup (Figure 1) and the steps needed to convert a unidirectional TTI to a bidirectional TTI using only off-the-shelf components. Second, we present RoMo, an open source Java library that allows existing applications to perform robot movement. We describe the design behind RoMo and demonstrate the functionality of the library with three applications that use robot movement to correct physical/digital conflicts. The robots and the RoMo library have previously

Submitted to DHRS2011

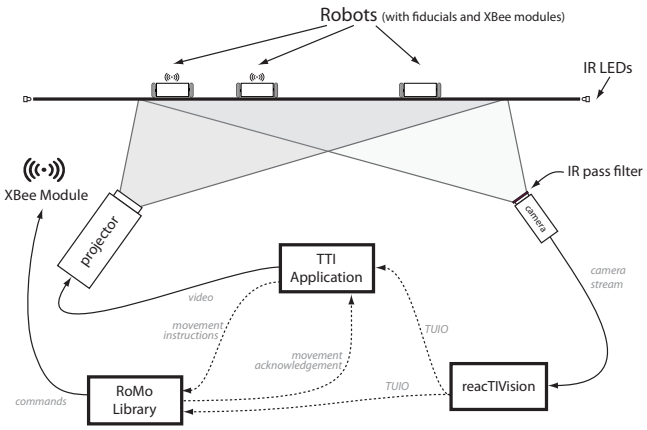


Figure 2. Overview of the system components.

been employed in [7] to study interaction with robotic tangibles.

RoMo HARDWARE DESCRIPTION

In this section we describe the hardware needed to use the RoMo library.

Tabletop

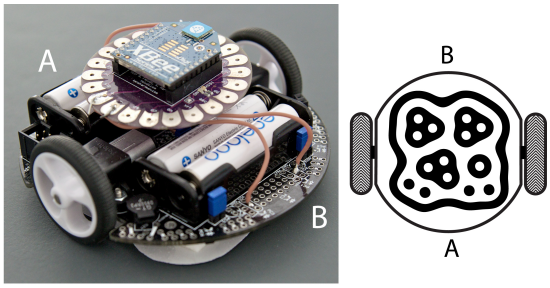
Figure 2 shows an overview of the components used in our tabletop setup. The setup is similar to that of unidirectional interfaces: Tangibles are marked with fiducials, which are illuminated by infrared light and tracked. The tracking information is sent as TUIO messages [4] to the TTI application where it is interpreted according to the interaction design of the application and where graphical visual feedback is generated. We use a DSI table [11] with reacTIVision [4] as tracking software, but any computer-vision based TUIO compatible setup will work.

The only hardware change made to the tabletop is the addition of a XBee Series 2.5 module to the hardware setup. The XBee module is mounted in an XBee Explorer USB-to-serial unit and is running as Communicator in API mode. This allows RoMo to communicate wirelessly with the robots at a speed of 115.2 kbps.

Robots

We use 3pi robots from Pololu [1] as tangibles (Figure 3). With a diameter of 9.5 cm and a weight of 83 g., they can easily be moved with only one hand. Moreover, the 3pi robots move and turn very quickly (100 cm/second). They are equipped with an Atmel ATmega328P micro controller running at 20 MHz and are programmable in C, Processing and Arduino. Each robot has a Lilypad XBee with a XBee series 2.5 module soldered on to the serial communication pins of the robot so as to allow wireless communication. The table in Figure 3 shows how the robot and the module are connected. A fiducial marker is attached to the bottom of the robot to make it visible to the tracking software. The fiducial marker must be oriented as shown in Figure 3.

[1] http://www.pololu.com/catalog/product/975



Lilypad	+	–	rx	tx
3pi robot	VCC	GND	PD1	PD0

Figure 3. A Pololu 3pi robot with a XBee module mounted in a Lilypad XBee. The table shows how the module is connected to the robot. A and B indicate how the fiducial should be oriented on the bottom of the robot.

MOTION IN RoMo

In RoMo robots are tracked with computer-vision and communication happens wirelessly. This introduces some challenges when controlling robots. In this section we explain how motion is achieved in RoMo.

Control and communication

Tracking and movement computations are performed solely by the control computer; the robots hold no positional information. The computer continously tracks the orientation and speed of each robot, determines the proper robot action and communicates this action to the robots in a round-robin fashion.

The steering of the robots is performed with robot movement frames. Each movement frame contains three values: (1) speed value for right motor, (2) speed value for left motor, and (3) movement time in milliseconds. The movement time helps avoid unwanted autonomous movement in case of package loss or package delay.

Rotation

Rotation is achieved by sending a pair of negated speed values to the robot. The value and the movement time are calculated from the angular distance to the destination angle; the bigger the difference, the higher the values. The maximum rotation speed is 220 deg/s.

Movement

Robot movement is done by driving in a straight line from the starting to the finishing position. In theory, this could be achieved by rotating the robot to the correct starting angle and driving straight until the finishing position has been reached. In practice, many challenges have to be addressed to make robot movement work. The two motors on the robots are not perfectly matched and at even motor speeds the robots pull to one side. Also, because of the kinetic energy stored in the motors, changes in motor speed are not immediately observable. Moreover, the latency caused by the tracking software and the wireless communication result

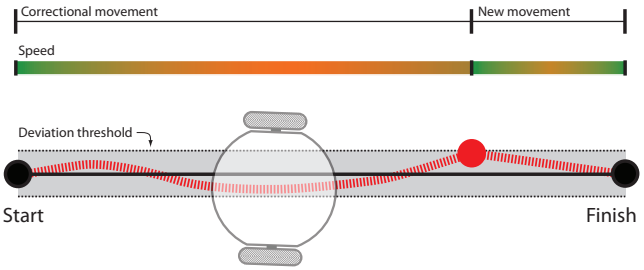


Figure 4. The robot's speed and movement direction is continuously adjusted to keep the robot on the path. If the robot exceeds the deviation threshold a new path is calculated and a new movement is initiated.

in a slight inconsistency between the actual robot position and the tracked robot position. Hardware challenges like the above mentioned are not specific to RoMo or bidirectional tangible interfaces in general; they are commonly discussed in the field of robot research, e.g. [2]. The robot control algorithm takes all these factors into account by performing constant tracking and adjustment of robot's speed and movement direction. In the following we describe our algorithm.

The robot controller initially calculates the equation for a straight line between the robot's starting position and its destination. The robot controller then rotates the robot to a position in which it is facing the destination and starts moving the robot at even left/right speed values. For each new position sample the robot controller receives, the deviation from the path is calculated. The deviation value is determined by calculating the minimum distance between the robot and the line. Small deviations are corrected by adjusting the motor speeds without stopping the robot. However, if the robot exceeds the deviation threshold, it is stopped, then a new path is calculated, and a new movement initiated. This results in a moving pattern similar to the one displayed in Figure 4. The deviation threshold and the maximum robot speed can be adjusted to achieve faster or more precise movement. As shown in Figure 4, the speed of the robot is adjusted depending on the distance to the finishing point. To ensure stable tracking and control, the robots move at a maximum speed of 24.5 cm/s.

THE RoMo LIBRARY

The RoMo java library consist of a small number of methods that TTI applications can use to initiate robot movement. In this section we cover the most important ones. RoMo is implemented to fit the observer design pattern commonly used in Java. The TTI application simply instructs RoMo to move a robot to a specific position or angle; when the desired destination has been reached, the TTI application is notified.

Rotation

One advantage of using robots in TTIs, compared to electromagnetic TTIs, is the ability to rotate tangibles. In RoMo robots can be rotated with the method `rotateRobot(int fID, float angleDegree)`, where `fID` is the fiducial id of the robot to be rotated and `angleDegree` specifies the angle ($0 \leq angleDegree < 360$) to which the robot should be rotated. With `angleDegree` set to 0, the robot is aligned parallel to the vertical edges of the screen facing the bottom edge.

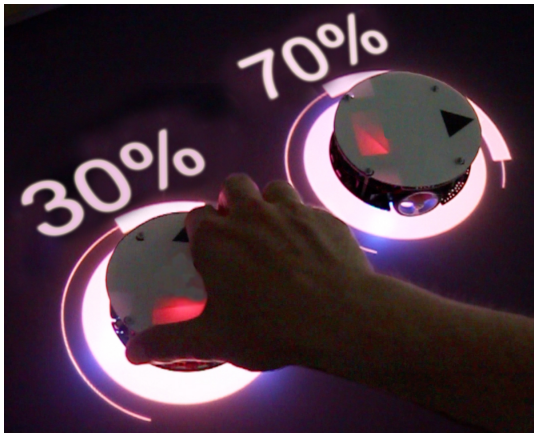


Figure 5. A user rotates a tangible to adjust the percent value. Physical/digital conflicts are avoided by rotating the other tangible, thereby preventing the total from exceeding 100%

Figure 5 shows an application that makes use of robot rotation to avoid physical/digital conflicts. Here, each robot is associated with a percent value displayed above the robots. The user adjusts the values by rotating the tangibles, using them as knobs. When the user rotates a robot to adjust the value, the other robot immediately rotates to the corresponding percent value, thereby ensuring a total percent value of 100%.

Movement

Robot can be moved to a new position by using the method `moveRobot(int fID, float x, float y)`, where `x` and `y` specify the coordinates to which the robot should be moved. `x` and `y` are expressed as percentage of the width and height of the screen ($0 \leq x, y \leq 1$). Figure 6 shows a simple application that uses robot movement to introduce constraints on the spatial layout of tangibles. In the application the user moves tangibles freely around the surface, and when the user stops interacting, the application forms an equilateral triangle by moving one tangible. The robot's movement path is displayed as a yellow dotted line.

The interaction design of this simple application illustrates an important difference between designing for unidirectional interfaces and bidirectional interfaces. In unidirectional interfaces the tangibles are only moved by the user and therefore tangibles can be used for interaction at any time. This is not possible when designing for bidirectional interfaces, as RoMo cannot differentiate autonomous robot movement from human intervention. If the user grabs and stops a moving robot, RoMo will interpret this as package loss and keep resending robot movement frames until the robot is finally at its finishing position. Consequently, the interaction design of applications for bidirectional interfaces will often need to

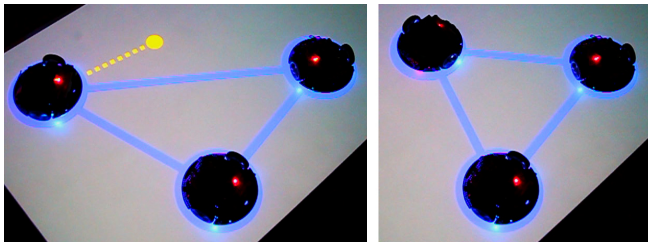


Figure 6. RoMo is used to introduce constrains on the spatial layout of tangibles. In this application robots are moved to ensure that the tangibles form an equilateral triangle.

Figure 7. A map application using RoMo to allow panning and rotation the map. Here, the user rotates the map by rotating a tangible.

employ turn taking and provide the user with feedback on when tangibles can be manipulated.

Figure 7 shows how RoMo may be used to allow interactions that would be impossible to employ in unidirectional tangible interfaces. In this application the user places tangibles on the map to mark certain sites (i.e., good places to dine). In unidirectional interfaces this would lock the view of the map, making it impossible to pan or zoom without creating physical/digital conflicts. However, in the bidirectional application shown in Figure 7, the robots move with the map, allowing the user to rotate the map by rotating a tangible or pan the map by moving one tangible. These interactions are normally only seen in multi-touch applications.

CONCLUSION

We have presented RoMo, an open source Java library that allows existing TTI applications to perform robot movement. We have described how unidirectional TTIs can be converted to bidirectional TTIs using only off-the-shelf components. In this paper we have focused on the use of RoMo to avoid physical/digital conflicts in TTIs. While we consider this an important quality of RoMo, we look forward to exploring and evaluating the novel interaction techniques that RoMo introduce to TTIs (i.e., undo of actions, recall of prior spatial layout and imitation of movement).

In our future research we plan on addressing the inability to detect human intervention by adding proximity sensors to the robots. Also, we will experiment with omnidirectional wheels to improve the robot's freedom of movement.

REFERENCES

1. Brave, S., Ishii, H., and Dahley, A. 1998. Tangible interfaces for remote collaboration and communication. In *Proc. CSCW 1998*. ACM, New York, 169-178.
2. Hogg, R., Rankin, A., Roumeliotis, S., McHenry, M., Helmick, D., Bergh, C., and Matthies., L. 2002. Algorithms and sensors for small robot path following. In *Proc. ICRA 2002*. IEEE. Washington, DC
3. Ishii, H. and Ullmer, B. 1997. Tangible bits: towards seamless interfaces between people, bits and atoms. In *Proc. CHI 2007*. ACM, New York, NY, 234-241.
4. Kaltenbrunner, M. 2009. reacTIVision and TUIO: a tangible tabletop toolkit. In *Proc. ITS 2009*. ACM, New York, 9-16.
5. Pangaro, G., Maynes-Aminzade, D., and Ishii, H. 2002. The actuated workbench: computer-controlled actuation in tabletop tangible interfaces. In *Proc. UIST 2002*. ACM, New York, 181-190.
6. Patten, J. and Ishii, H. 2007. Mechanical constraints as computational constraints in tabletop tangible interfaces. In *Proc. CHI 2007*. ACM, New York, 809-818.
7. Pedersen, E. and Hornbæk, K., 2011. Tangible Bots: Interaction with Active Tangibles in Tabletop Interfaces. In *Proc. CHI 2011*. ACM, New York. Anonymized
8. Ressler, S., Antonishek, B., Wang, Q., and Godil, A. 2001. Integrating active tangible devices with a synthetic environment for collaborative engineering. In *Proc. Web3D 2001*. ACM, New York, 93-100.
9. Richter, J., Thomas, B. H., Sugimoto, M., and Inami, M. 2007. Remote active tangible interactions. In *Proc. TEI 2007*. ACM, New York, 39-42.
10. Rosenfeld, D., Zawadzki, M., Sudol, J., Perlin, K. 2004. Physical Objects as Bidirectional User Interface Elements. In *IEEE Comput. Graph. Appl.* (Vol. 24, no. 1), 44-49.
11. Schöning, J., Brandl, P., Daiber, F., Echtler, F., Hilliges, O., Hook, J., Löchtefeld, M., Motamedi, N., Muller, L., Olivier, P., Roth, T. and Zadow, U. von Schöning. 2008. Multi-touch surfaces: A technical guide. In *Technical Report TUM-I0833*, Technical Reports of the Technical University of Munich, (October 2008)
12. Sugimoto, M., Kagotani, G., Kojima, M., Nii, H., Nakamura, A., and Inami, M. 2005. Augmented coliseum: display-based computing for augmented reality inspiration computing robot. In *Proc. SIGGRAPH 2005*. ACM, New York.